DATA ENGINES FOR VERSION 6 OF THE SAS® SYSTEM UNDER VMS™

Darylene Colbert, SAS Institute Inc., Cary NC

ABSTRACT

This paper discusses the data library engines that are available under VMS™ for Version 6 of the SAS® System. A discussion of engines in general is presented, along with a discussion of the specific engines available to the VMS user. Assigning engines to libraries and examples of using engines are described. This paper also discusses engine compatibility issues and future directions of engine design for VMS.

INTRODUCTION

In Version 6 of the SAS System, access to data has been enhanced through implementation of the Multiple Engine Architecture (MEA). The MEA enables the SAS System to process any file type as if it were a SAS data set. This paper does not detail the internals of the MEA; however, it does explain how the user can access various file types by using engines that are provided as part of the MEA.

This paper is divided into four sections. The first section describes engines in general — how to access files using engines, how to access directories containing data libraries from multiple engines, and how to control creation and deletion of the WORK library. The second section discusses native engines under VMS. Native engines are engines that access any form of a SAS file that is created and maintained by SAS Institute Inc. In addition to native engines, Version 6 of the SAS System supports a variety of interface engines. An interface engine reads and/or writes other file types (that is, non-SAS files) as if they were SAS data sets. Section three discusses interface engines available under VMS. The last section briefly discusses future plans for engines.

I. DATA LIBRARY ENGINES

The set of routines used to access a file as if it were a SAS data set is referred to as an engine. One could think of an engine as a data translator. An application makes calls to an engine to read or write data. The engine then calls host-specific code to read or write the data. To the application, the data and metadata are presented in a way that is expected of SAS data sets. In the host-dependent code, the file is read or written in its own format. The engine, therefore, performs the translation between the physical format and the format of a SAS data set.

Engines are useful because they provide access to many different file formats directly; you do not have to physically convert the file to a Version 6 SAS data set for it to be used by SAS procedures. Because of this, the SAS System can access many data formats other than standard SAS data sets directly. For example, in Version 6, the format of a SAS data set has been changed, making compatibility a major concern. The MEA allows the SAS System to support compatibility engines which can transparently access Version 5 SAS data sets.

The MEA is also flexible enough to allow additional engines to be added to the SAS System already in the field or to implement new engines for new data formats from other vendors. Since there is a standard engine interface, at some point in the future, it will be possible for users and vendors to write their own engines.

ASSOCIATING ENGINES AND LIBRARIES

To use an engine, you must associate a specific engine with a particular library. Each library has only one engine assigned to it. There are two ways to accomplish this. You can either use the LIBNAME statement to make this assignment explicitly or you can let the SAS System default to a specific engine to use implicitly.

Explicit Assignment of an Engine Name

You can explicitly assign an engine name to a library by using the LIBNAME statement. In Version 6, the LIBNAME statement has several new options available. One of these is the name of the engine to be associated with this library. The general form of the LIBNAME statement in Version 6 is

LIBNAME *libref <engine> <physical name> <engine options>*;

where

libref	is a SAS name that identifies this library.
engine	is a SAS name that specifies the engine to assign to this library (for example, V606, COMPAT, V6RMSSHR).
physical name	is a quoted string that specifies the physical location of this library. Under VMS, this is a directory specification.
engine options	are engine options that are to be applied to this library.

For example, if you want to assign the library TEST, which resides in [CURDIR], to the V5 engine, you use

```
LIBNAME TEST V5 '[CURDIR]';
```

You then access Version 5 SAS data sets in that library just as you always have by specifying the libref. For example, to print data set MYDATA in the library that we just defined, you type

```
PROC PRINT DATA=TEST.MYDATA;
RUN;
```

Implicit Assignment of an Engine Name

If you do not specify an engine name in the LIBNAME statement, an engine is implicitly assigned to the given libref. This implicit assignment occurs either at the time the LIBNAME statement specifies the libref without an engine name or when the libref is referenced for the first time if there was no LIBNAME statement at all.

The SAS System under VMS attempts to determine the engine that should be assigned to the given libref by looking at the file extensions of the files in the directory specified. If there are SAS files from only one of the native engines supported by SAS Institute Inc., the libref is assigned to that engine. If there are no SAS files in the given directory or there are SAS files from more than one native engine, the libref is assigned to the default engine. For Release 6.06, the default engine is V606. However, this can be changed with the DEFENG= option.

If you explicitly specify an engine name to be assigned to a libref, the SAS System does not have to take the time to look at the files in the directory to determine the engine to use on your behalf. Therefore, for production jobs, it is beneficial to use an explicit assignment. You also ensure that the desired engine is used to reference the data library. However, many times it is desirable to continue using the data file format that already exists in a particular library, whatever engine that happens to correspond to. In these cases, let the SAS System implicitly assign the engine on your behalf.

As mentioned above, in Version 6 of the SAS System under VMS, you can reference a libref without issuing a LIBNAME statement. This is different from Version 5. When the SAS System encounters a libref

that has no corresponding LIBNAME statement, it assumes that the libref refers to a VMS logical name. Obviously, when you use a VMS logical name without having used a LIBNAME statement, you cannot specify an engine name. Therefore, the engine name is implicitly assigned to the libref using the rules stated above. To use a VMS logical name as a libref without using a LIBNAME statement, you must define the logical name externally to the SAS System as shown here:

```
$DEFINE LOGICAL [MYDIR]
```

After you enter the SAS System, you can use LOGICAL just as you would a libref that has been assigned using a LIBNAME statement.

```
DATA LOGICAL.FIRST;
  SET LOGICAL.SECOND;
RUN;
```

If the directory [MYDIR] has only Version 5 data sets in it, the libref LOGICAL is implicitly assigned to the V5 engine when the DATA statement is executed and data set FIRST is created as a Version 5 SAS data set.

MIXED MODE DIRECTORIES

When there are SAS data sets from more than one native engine in a single directory, this is called a *mixed mode* directory. For example, if you have Version 5 SAS data sets and Version 6 SAS data sets in the same directory, then you have a mixed mode directory. This is not recommended. The SAS System is able to distinguish the different files by their file extensions; however, it could become quite confusing for the user. The SAS System will never create a mixed mode directory by default. You would have to explicitly assign a different engine to the library and create data sets in it for the directory to become mixed.

If you have a mixed mode directory, for example one consisting of Version 5 and Version 6 data sets, the libref that is assigned to the Version 6 engine is able to access only the Version 6 data sets. The libref that is assigned to the Version 5 engine is able to access only the Version 5 data sets. To see how confusing this can become, consider this example.

The directory [MYDIR] contains two SAS data sets as shown by the following VMS DIRECTORY command:

```
$DIR

Directory DISK1:[MYDIR]

A.SASIO604$DATA                A.SSD

Total of 2 files.
```

One of these data sets is in Version 6 format and the other is in Version 5 format.

```
LIBNAME NEW V606 '[MYDIR]';
LIBNAME OLD V5  '[MYDIR]';

PROC PRINT DATA=NEW.A;
RUN;

PROC PRINT DATA=OLD.A;
RUN;
```

In this example, you have two data sets with member name A that reside in the same directory. The first PRINT procedure prints the Version 6 data set A (file A.SASIO604$DATA). The second PROC PRINT prints the Version 5 data set A (file A.SSD). As you can see, this could become hard to read and understand in your SAS programs. Therefore, it is recommended that you create data sets from only one engine in a single directory.

THE WORK LIBRARY

The only temporary library available in the SAS System under VMS is the WORK library. All scratch files used by the SAS System are created in the WORK library. On a VMS system, the WORK library is implemented as a subdirectory. By default, it is a subdirectory of the directory assigned to the logical name SAS$WORKROOT. At the end of the SAS session, all files in the WORK library are deleted and the subdirectory is deleted. Therefore, any SAS data sets that you create in the WORK library are temporary and are deleted at the end of the session. By default, data sets that do not explicitly specify a libref (for example, one-level data set names) are in the WORK library. The WORK library is always implicitly assigned to the base engine. The base engine is the native engine that supports the current release's SAS data set format. For example, for Release 6.06, the base engine is V606.

You can change the location where the WORK library is created by using the WORK= option. Unlike Version 5, this option is available only on the command line. Therefore, you have only one WORK library for each SAS session in Version 6. By default, the WORK library is created under your current directory. To change the location where the WORK library is created to a scratch location, you specify

```
$SAS/WORK=SCR:[SCRATCH]
```

In this example, the subdirectory is created as a subdirectory of SCR:[SCRATCH]. Note that the WORK= option is no longer available as a SAS global option within a SAS program.

There are also two options available to control the creation and deletion of the WORK library. By default, the WORK library is created at the beginning of the SAS session and deleted at the end of the session. The WRKINIT option controls the creation of the subdirectory. The default is WRKINIT. If you specify NOWRKINIT, the SAS System looks for an existing WORK library in the current directory or in the directory specified by the WORK= option and uses it, as is, if it exists. If it does not exist, then it creates one. WRKINIT is a command-line-only option.

The ERASE or WORKTERM option controls whether the WORK library is deleted at the end of the SAS session. ERASE and WORKTERM are synonyms. ERASE is used on the command line, whereas either ERASE or WORKTERM can be used in a SAS program as a global SAS option. The default is ERASE or WORKTERM. If you specify NOERASE or NOWORKTERM, the WORK library is not deleted at the end of the session and is available for use with the next SAS session. Remember, however, that you must specify NOWRKINIT in the next invocation of SAS to reuse the WORK library that you saved.

II. NATIVE ENGINES

A native engine is an engine that accesses formats of SAS files that are created and maintained by SAS Institute Inc. Examples of native engines are the base engine and the Version 5 compatibility engine. Both of these engines access SAS files, but the files are read and written using different formats. The native engines and the engine names for Release 6.06 of the SAS System under VMS are

- Base — V606 or BASE
- Compatibility — V5 or COMPAT
- Sequential — TAPE or SEQUENT
- Transport — XPORT
- View — SQLVIEW
- Concurrency — V6RMSSHR

BASE ENGINE

The base engine is the default engine for a particular release of the SAS System. For Release 6.06, the base engine is V606. You also can reference this engine as BASE. If you use BASE as the engine name, then in future releases of the SAS System, you will always be using the engine that was shipped as the default for that release. However, if you use V606, you will always be using the Release 6.06 engine regardless of what version of the SAS System you are running. If you are creating new data sets and want to use the latest engine, then you should use the engine name BASE. Otherwise, it is recommended that you use V606.

The base engine for Release 6.06, V606, introduces quite a few new data management features such as indexing, where clause processing, and data set compression.

COMPATIBILITY ENGINE

The compatibility engine is used to access Version 5 SAS data sets. The compatibility engine can be assigned using the names V5 or COMPAT which are synonyms for the engine. One of the biggest benefits of the compatibility engine is that it allows the user to continue to access Version 5 SAS data sets without changing the syntax of current programs. Due to the implicit assigning of engine names to librefs in Version 6, the compatibility engine is assigned to a library that currently contains only Version 5 SAS data sets. Therefore, the program is able to continue to read and write Version 5 SAS files with no change to the LIBNAME statement.

This greatly eases the transition from Version 5 to Version 6 of the SAS System. While you are still encouraged to convert data libraries to Version 6 format to take advantage of new features such as indexing and data set compression, this conversion can take place at your convenience.

SEQUENTIAL ENGINE

The sequential engine is provided to create SAS data sets in a format that can be used on sequential devices. Sequential data sets have a much simpler format than data sets created by the base engine because no random access is allowed. As in Version 5, sequential SAS data sets are used typically to store SAS data sets on tape. However, this format is not limited to tape. The sequential engine is referenced using the name TAPE or SEQUENT.

TRANSPORT ENGINE

The Version 5 transport format of a data set is supported in Version 6 through the transport engine. The transport engine, named XPORT, accesses a machine-independent file format that can be created or read on any machine running Version 5 or later of the SAS System. In Version 5 of the SAS System under VMS, the TRANSPORT= option was used to read or write transport format data sets. You also used a FILENAME statement to specify the name of the transport library. In Version 6, since transport data sets are created using an engine, a transport library is referenced by specifying the XPORT engine in a LIBNAME statement. The TRANSPORT= option is obsolete.

Prior to Version 6, different syntax was required on different operating systems to access transport data sets. The XCOPY procedure was used under CMS, MVS, and DOS/VSE, while the TRANSPORT= option was used for the minicomputers. The transport engine now provides a common syntax for all hosts running Version 6 of the SAS System.

VIEW ENGINE

The SQL procedure, which is new for Version 6, creates and stores views defined by SQL operations. A view can subset a SAS data set, or it may be a logical concatenation of any number of SAS data sets or subsets of SAS data sets. The View engine, or SQLVIEW, uses these views to access data in subsequent procedures or data steps.

CONCURRENCY ENGINE

Many times, multiple users need simultaneous access to SAS data sets. For example, several people may want to update the same SAS data set using the FSEDIT procedure at the same time. In previous releases under VMS, this was not possible. However, in Version 6 of the SAS System under VMS only, there is an engine that provides this capability. It is called the concurrency engine. Its engine name is V6RMSSHR. By using the concurrency engine, you can access a single SAS data set from multiple SAS sessions on the same machine or within a VAXcluster.

This engine creates and accesses SAS data sets that are in an acceptable format to allow VAX RMS™ record locking and file sharing. They are not interchangeable with SAS data sets created with any other engine. This is true of all engines; each engine uses a distinct format.

If you plan to share a particular SAS data set, you should create it using the concurrency engine. Of course, you can always change the format of an existing SAS data set by copying it and specifying an output engine that is different from the input engine. For example, if you have a data set MYDATA that was created using the base engine and you decide you would like shared update access to this data set, you could use the following syntax to convert it:

```
LIBNAME INLIB '[MYDIR.BASE]';
LIBNAME OUTLIB V6RMSSHR '[MYDIR.SHARE]';
PROC COPY IN = INLIB OUT = OUTLIB;
  SELECT MYDATA;
RUN;
```

After running this SAS program, the data set MYDATA that is created with the base engine in INLIB is copied to OUTLIB using the concurrency engine.

Concurrent access is most important when multiple users want to update a single SAS data set. The FSEDIT procedure is the only procedure that allows updating of SAS data sets.

III. INTERFACE ENGINES

Along with the native engines, Version 6 of the SAS System supports a number of interface engines. An interface engine is one that reads and/or writes another vendor's file formats, that is, non-SAS files. The engine presents the data to the SAS System as if it were a SAS data set. These engines provide tremendous growth potential for the number of different types of file formats that the SAS System will be able to process in the future.

Currently planned for Release 6.06 of the SAS System under VMS are two interface engines — an interface to ORACLE® and an interface to VAX Rdb/VMS™. Both of these interface engines are used in conjunction with a new procedure, the ACCESS procedure. By using PROC ACCESS, the user first extracts the definition of a relation or table from the database into a master definition. The user can then create multiple views using this master definition. The database is accessed through the created views, and the actual data is extracted at procedure execution time.

A user can read data from a relation or table using any SAS procedure. He or she can also update, insert, or delete data using PROC FSEDIT. However, for Release 6.06, there will be a few limitations on access to the databases. The user cannot create a new table or relation from within the SAS System. Also, views that involve more than one table or relation cannot be created.

IV. FUTURES

As you have seen, Release 6.06 of the SAS System will allow direct access to many types of data files. However, we intend for this list to keep growing. One of the best ways for this list to expand is to provide

a way for users and vendors to write their own engines. The engine interface specifications will be published in a technical report to provide the information necessary to write an engine, just as users can now write their own procedures to be used within the SAS System.

There are a number of interface engines planned to be implemented by SAS Institute Inc. Access to VAX RMS files whose field definitions are stored in the VAX Common Data Dictionary is one of the first areas to be explored. Access to IBM® databases through the VIDA™ with IDMS/R™ link is certain to be addressed. The DIGITAL™ Compound Document Architecture (CDA)™ is another area that could be an important interface to the SAS System.

New native engines that will be coming include the remote engine. The remote engine is a communications link that interfaces with a detached server to allow the exchange of data within SAS/SHARE® software. This engine uses host-dependent communications software to communicate requests for data to this detached server. It allows the sharing of SAS files across different hosts or machines.

CONCLUSION

SAS Institute Inc. has provided a mechanism for users to combine the power of the SAS System with many different types and formats of data easily. The architecture provides for expandability by SAS Institute Inc. as well as users and vendors. It also provides compatibility with Version 5 and conversion capabilities.

If there are data formats under VMS to which you feel SAS Institute Inc. should support an interface, please use the SASware Ballot® to express your desires or talk to one of the VMS developers during the SUGI conference.

SAS, SAS/SHARE, and SASware Ballot are registered trademarks of SAS Institute Inc., Cary, NC, USA.
VMS, VAX Rdb/VMS, DIGITAL, CDA, VAX RMS, and VIDA are trademarks of Digital Equipment Corporation.
IDMS/R is a trademark of Cullinet Software, Inc.
ORACLE is a registered trademark of Oracle Corporation.
IBM is a registered trademark of International Business Machines Corporation.